

G HARDWARE, SOFTWARE AND ALTERNATIVE ARCHITECTURE CONSIDERATIONS

G.1 INTRODUCTION

The purpose of this appendix is to present the alternatives and considerations associated with hardware and software, as well as several technical architectures that can be useful in the WIC environment. It is the intent of this appendix to discuss at a high level the architectures that can be considered using current infrastructure technologies (CPUs, databases, operating systems, network operating systems, etc.). The requirements that drive the selection of a particular architecture are also discussed.

G.2 EQUIPMENT AND FACILITIES

The WIC system described in this document requires different basic hardware configurations, depending upon whether the system uses centralized or distributed processing. For centralized processing, the basic configuration requires no special equipment. Included in the basic hardware configuration are:

- Central Processing Unit (including main memory)
- Direct Access Storage Devices
- Tape/Cartridge Drives
- Terminals
- Printers
- Telecommunications Hardware

If an agency chooses to process returned checks and vouchers (instead of contracting for this service with a bank or processor), it will require special check processing hardware designed to read Magnetic Ink Character Recognition (MICR) encoded instruments.

The size and speed of the main computer is dependent on workload projections, anticipated caseload, number of local agencies supported, transaction volume, desired response times, growth rate, and other variables. There is no simple

sizing algorithm available for agencies to use to determine the optimal system size. Most hardware vendors provide these algorithms. In light of the program growth rate, it is critical for agencies to plan for a hardware contract and configuration that will support growth. In addition to participation growth factors, State agencies need to consider transaction growth when determining optimal hardware configurations. The number of functions performed by the State agency's automated system and the number of users performing processes during peak periods will affect the overall processing speed and capability of the system, as well as storage and memory requirements.

If, on the other hand, an agency opts for a distributed processing configuration, personal computers and/or minicomputers will be required for local agency processing. For the purpose of this FRED, it is not possible to size the overall hardware configuration. The storage and capacity of each computer installation is highly dependent on the WIC Program volume and the number of WIC functions to be supported. For example, some small local agencies will utilize only a few PCs to support all WIC functions with relatively low volume. Other large agencies will have higher volume, and may have some computers dedicated to a single function, such as enrollment intake or food instrument production. It should be noted that a number of States that are developing distributed WIC systems are utilizing local area networks (LANs) to link personal computers and share data at the local agency level, and also at the State central office. Further, a few clinics located in remote areas without convenient access to telecommunications lines are using wireless networks to communicate with the central processing site.

As States move progressively toward EBT/ESD solutions, additional equipment will be required. For the retailer, the adoption of EBT will require the following types of equipment:

- Scanning System. Ideally, when the food purchase takes place, the retailer system should have a scanner, which is integrated with the retailer's electronic payment system (EPS).
- Electronic Payment System (EPS). The retailer needs a computer to store the UPC database so that WIC food item purchases are screened and to capture the transaction data. A computer is also

needed to execute the transmission of the vendor transaction detail file for ACH payment and to the EBT Processor. This capability should be integrated with electronic cash register functionality.

- Point of Sale Terminals. The terminal must process multiple item transactions and route food item data back and forth with the retailer EPS, card, and in an on-line system, the EBT Processor. The POS devices may be able to read magnetic stripe cards, smart cards, or both.
- Administrative Terminal With Printer. In the EBT environment, most retailers have an administrative terminal with a printer that is used to print the participant's shopping list.

WIC clinics participating in EBT/ESD will need cards and card readers. Provider offices will require card acceptance devices, at a minimum, as well as telecommunications capabilities to be able to transmit transactions to a centralized host or through the Internet. Wireless technology or the Internet may provide a solution for remote providers who do not have convenient access to telecommunication facilities. Those agencies issuing EBT/ESD cards will require card personalization and issuance workstations. Depending on the type of technology being implemented, EBT/ESD may also require supplemental devices to capture digitized signatures and/or photos, as well as live biometric scans. For programs using digital signatures, secure key pair generating workstations may be needed, if Public Key Infrastructure (PKI) services are not outsourced as well.

Participant confidentiality is an issue with all information systems that deal with public information. The method by which confidentiality is ensured is dependent upon the standards and guidelines for confidentiality. The implementation of security controls and limiting access to the system is a subset of the methods to be implemented for ensuring confidentiality. It is essential to ensure that only authorized personnel are able to gain access to personal enrollee information, to produce food benefits, and perform other sensitive system functions. It is likewise essential to ensure that WIC hardware and software is adequately safeguarded to minimize the possibility of program interruption. Adequate security is a particular concern for local WIC agencies that have limited or no experience in providing security for a critical automated system. State and local agencies should use passwords to restrict system access and protected data fields

that can only be changed with a supervisor's authorization. Disaster planning is important as well. Off-site storage of important system data files is one example of what should be considered as part of disaster planning.

Several emerging technologies hold promise for improving security within the WIC system. In the future, smart cards carrying digital certificates can be used to securely authenticate the identity of a user of the WIC system. Alternatively, passwords could be stored on the card. Biometrics provides yet another alternative to securely proving both user and participant identity. Biometrics could be used to secure access to the system as well as to control access to private health data in the clinic setting. As the WIC Program moves toward the use of Internet and kiosk technology, the ability to authenticate an identity for a remote user will become more and more important.

Another area for States to consider is the use of emerging technologies for the improvement of WIC service delivery. Appendix D discusses several technologies that effectively can be used by State WIC Programs. However, given state budgets, cost could be a significant factor in mitigating against the adoption of kiosks, data warehousing, geographic information systems, web-applications, and automated response units. When employing the following technologies, specialized equipment may be required:

- Data warehousing, which is the technology of storing data in a relational database in a manner conducive to performing data analysis and ad-hoc queries.
- Wireless technology, which within the context of a WIC application, refers to the following capabilities:
 - Connecting PC clients to a server using infrared or radio frequency technology. This technology would typically be used in a building where it is not cost-effective or physically possible to implement a LAN wire infrastructure.
 - Connecting a PC client to a server using cellular technology. This technology would typically be used in mobile clinics or remote clinics where access to the server is not available through a standard LAN or dial-up technology.
- Email and World Wide Web Communication System, which are ubiquitous terms for applications that ride upon the Internet infrastructure, specifically for the sharing of data and information. Within the context of the WIC Program, the Internet could be used to

transport payment/reconciliation data, retailer sales data, or back-up transactions. World Wide Web applications could be used to perform electronic vendor certification or participant applications.

- Use of Electronic Signature Capture to provide an alternate means of verification of acceptance or acquiescence conveyed by a customer signature.

Another concern for States, when implementing an automated system is system integration or linkage with other programs within the State and across States. Data processing systems can no longer operate in a vacuum. Data sharing is an essential part of doing business in today's world. In many cases, the linkage with other programs is not known when the requirements for the base system are being defined. An example of this is the linkage with an EBT system. The basic data elements can be identified, but exactly how the WIC system will integrate with an outside vendor's EBT system will not be known until an EBT vendor is selected and requirements are defined. Consequently the key is to define system requirements that allow the base system to be modifiable and extensible to other programs. Then as additional linkage and integration with other programs are required, it can be accomplished with a minimal of effort.

While interfaces to legacy systems provide one alternative for sharing data among cooperating programs, emerging technologies offer two additional approaches to secure data sharing: network-based and card-based. Both concepts could be useful to WIC agencies in different circumstances, depending on the environment and the requirements of a particular program. For example, while some agencies have well-established network-based systems and would like to link these with other programs' systems, other agencies have a particular need for a portable, off-line information carrier that is viable when telecommunications are not available. For those situations, the smart card can be used to create linkages across disparate systems without the creation of system specific interfaces.

The network approach relies upon the concept of a "virtual" patient account maintained on the Internet. The concept of a virtual patient account – pulling information from multiple databases and displaying it through a web-based application as a consolidated patient record – could enable sharing of data by

various health programs in both the public and private sectors. The card could carry demographic information, a digital certificate, and the programs in which the card holder participates. The card could be read, and once the card holder's identity and right to access data is ensured, the application would pull health/medical information from multiple public program and/or private provider databases (e.g., WIC, Head Start, Immunization Registry, Maternal and Child Health, HMOs etc.) and present it through a browser-based application. Therefore, up-to-date information from each previous provider, whether public or private, would be presented to the current provider querying the card holder's virtual health "account" to obtain necessary information to better serve the needs of his/her patient.

Whatever approach is ultimately used for data sharing by the WIC Program, it is clear that this need will become increasingly important in the future as government agencies at the Federal and State levels move ever closer to the electronic delivery of services to their citizens.

G.3 SOFTWARE

There are two types of support software – system support software and application support software. System support software includes the operating system, system utilities, performance monitoring software, etc. Application support software includes data base management systems, report writers, statistical and graphics software packages, and other such software designed to support or work in concert with the application software. Additionally, centralized technical support should be available to all software and application issues.

The operating system for the main central processing unit (CPU) does not require any special features above those found in conventional mainframe operating systems. It should be capable of concurrently supporting local batch and interactive processing in a multiprogramming mode. It should have integrated control over all system and application software, all system I/O functions, and all interfaces with local peripherals, communications interface, and communications subsystem. It should protect all system software, application software, and data files. It should monitor and control concurrently executing jobs co-resident in

the main memory. In a distributed environment, a networked operating system (e.g., Windows NT, Unix, etc.) will generally be required.

System utilities perform functions necessary to facilitate the operating system and application software. System utilities mainly perform file and volume (tape, disk) handling functions. The WIC system requires basic system utilities.

Utilities should be supplied to copy and back-up files and volumes, initialize tapes, list tape label information and file characteristics, and copy tape volumes. Direct Access Storage Device (DASD) utilities should initialize volumes, label and re-label volumes, search volumes, copy volumes, etc. Utilities should be provided to list data file names, characteristics, space requested and used, and dates created and last used. Utilities should be provided to sort and merge data files and, for distributed processing systems, to support overnight uploading and downloading of data that is initiated by a centralized processing site.

While the use of the system support software is required, use of application support software is optional. State agencies will also have to make decisions regarding the application software to use on its automated system. While use of database management software was only recommended in the last version of the FRED, it is now commonplace. When choosing software for their systems, States should be cognizant of open platform standards. For example, the question today is not whether to employ a database management product, but rather what products are compliant with the Open Data Base Connectivity (ODBC) standards.

There are numerous commercial over-the-shelf (COTS) software packages that will meet a State's needs. In some States, all required functionality might not be available through the main WIC IS. Rather, commercial software packages running on PC's and using data extracted from the WIC IS may provide a more efficient alternative for some functionality. COTS spreadsheets, for example, may be particularly effective for caseload management and various forms of financial modeling. Calendaring packages may be equally viable for providing scheduling functionality, and generic financial management software may provide necessary financial monitoring. Groupware and communications packages can be used to streamline the exchange of information between State

and local agencies. Commercial report writers, ad hoc query, statistical analyses, and on-line analytical processing (OLAP) tools can provide convenient and cost effective reporting capability. Agencies are strongly encouraged to assess the costs and benefits of database management systems, report writers, statistical analysis software, and data management software (e.g., Data Expert) for use in a WIC system.

As the WIC Program moves further into the future, software related to the use of the Internet will become progressively more important. Web-browsers, electronic mail, and Internet Service Provider access software will become necessary, as will software to deliver improved security. Software to encrypt transactions and to verify the identity of users seeking remote access to the system will be needed as WIC transactions increasingly move to the Internet. Further, as EBT/ESD becomes more commonplace, card management software will be needed, if this function is not outsourced.

G.4 TECHNICAL ARCHITECTURE ALTERNATIVES

This section presents, at an overview level, several architectures along with the major considerations that generally affect the choice of one over the others. It is not intended to be an exhaustive study of all the potential architectures nor a detailed argument for or against all the variations of each. Note that in at least some cases, it is possible to combine architectures, building an overall system that uses a “best of breed” approach to solve multiple, concurrent problems.

States may implement an automated WIC system in numerous ways and can make use of myriad technologies to deliver services. For example, a State may choose to develop a centralized, on-line system in which data is transferred between WIC clinics and a central database at the State agency on a real-time basis. Another State may find that a distributed database system better meets its needs. These different system architectures are highlighted in Appendix E, which presents a comparison of several State WIC systems. States moving toward WIC automation will have to make decisions about what kind of system they will implement based on numerous factors such as WIC caseload size, geographic distribution, cost, telecommunications infrastructure, and technical support for hardware problems/issues among others.

G.4.1 WHAT IS AN “ARCHITECTURE?”

By the term “Architecture,” we mean the fundamental organization of the system. There are a number of ways to classify system architectures. The method used here will be to divide systems into Centralized vs. Distributed (referring to data and programs separately) as well as Client/Server vs. a single, large system (sometimes referred to as a monolithic system). Since these terms are frequently used without a formal or standardized definition, the following definitions will be used here solely for the sake of clarity.

- Centralized – The program(s) and/or data are primarily contained and used on a single computer or bank of computers acting as a single unit.
- Distributed – The program(s) and/or data are contained on multiple computers that do not act as a single unit. Those computers may execute different parts of the application and/or store different parts of the data, or they may have the same data and/or programs and execute in a parallel mode.
- Client/Server – The system is divided into two primary types of functions. The “Client” functions are primarily concerned with user interface, presentation of data, and manipulation of data. As needed, they will request services from one or more “Server” components of the system. (The users may be human or may be other computer systems or programs.) The “Server” functions provide one or more services to a number, but not necessarily to all, of the “Client” processes in the system. Note that all clients and/or all servers could exist on one computer or on some combination of many computers. Client/Server refers to the distribution of tasks within the system, and not to the physical location of the various tasks. C/S can be Centralized, Distributed, or a hybrid. The most common “Client” elements are individual computers concerned with data manipulation. The most common servers are database servers providing selected information to clients and storing the results of client manipulation of data.

Most C/S systems are dispersed over multiple computers. In all cases where there is more than one computer involved, the network linking Clients and Servers is an integral part of the architecture, not a separate and distinct issue. (The physical implementation may or may not be distinct, but the logical network is integral to the system. It is sometimes stated that, in C/S, “the network is the computer.” This heuristic is less than fully accurate, but serves to emphasize a key point.)

- Monolithic – The system is not divided into “Client” and “Service” modules. All interfaces and functional processes are normally

grouped together without regard to classification other than for library organizational purposes. Until recently, most computer systems were organized in this manner. These systems may be either centralized, distributed, or a hybrid.

This document will remain at an overview level. It will not explore subset options within these larger architectural genres such as “3-tier v. 2-tier client server.”

In addition, architectures are usually concerned with whether the supporting infrastructure will be “mainframe” or “microprocessor” based. In general, so-called “mid-range” computers can generally be used as either “mainframes” or “micros,” as power and cost needs dictate. Either type of computer hardware architecture can be used in any combination in any of the noted systems architectures. For the most part, the decision is based on the history of the organization, the computers already in existence at the organization, costs, and personal preferences of those choosing the systems infrastructure.

There are a few areas where one hardware technology dramatically outshines another, such as mainframes or even “supercomputers” for very-large-scale, multidimensional array processing. To be reasonable, any cost comparison of hardware architectures must include “life span of system” costs rather than just acquisition costs. In general, mainframes and mid-range systems have higher initial acquisition costs and lower on-going support and maintenance costs for the same application. Microprocessor systems are generally the opposite, with deceptively low initial acquisition costs followed by much higher support and maintenance costs. This is aggravated by the extremely fast obsolescence of microprocessor software, which further drives up the long-term costs.

G.4.2 HISTORICAL OVERVIEW

When one looks at older, real-world database systems, often their “design” was driven by narrowly focused information requirements. Usually, the database, and the entire system was designed to support a narrowly defined and specific set of reports or similar outputs. In these older systems, while some amount of time (albeit small) may have been spent discussing possible future changes, or even incorporating specific future changes, little real design effort went into ensuring

flexibility to meet undefined future needs. As the need for information grew, the database and the application were modified to include the additional data.

As such, this kind of database system can be compared to a house being built one room at a time. Each new room was added on—somewhere that seemed convenient, or at least not too inconvenient—as the need arose. Over time, the structure became more and more unwieldy and more difficult to modify further. To carry the analogy further, over time there would be no more room for horizontal expansion, so new rooms would be added upward. Of course, the underlying structure was not intended to support second and third stories. Even with bracing and shoring up of the supports, it eventually became impossible to add any more rooms without causing the whole thing to come crashing down.

Today's information system requirements are much more complex and too critical to be done by this approach. And the definition of what is the system is bigger. Not only is it a database system, it is a system that includes the database engine and application, but it also includes communications and interfaces to other database systems. The users of the system are no longer just a small, predefined set of humans. The users will often be people that were not even identified at the time the system was designed or built. Some of those users may not even be humans, but instead may be other computer systems.

All of this requires not only new data, but whole new interfaces be appended to the system without causing it to crash under its own unwieldy weight. Today's systems need to represent and reflect the notion that it must fit within an information system that serves the users' objectives and meets its operational needs. The database by itself and the system in a larger sense must serve strategic as well as tactical objectives. More importantly, it must have the inherent flexibility to accommodate growth and other change.

A WIC application by its very nature, as reflected in its requirements, is a database system with a complex set of functions that cover a wide scope of tasks that often seem distantly related, if related at all. In some cases, the only real relationship between tasks is that they are both part of providing WIC services or are both part of the required reporting. Over time, the WIC program changes in a

non-trivial fashion. The rules change. The technical and administrative practices and requirements change. The environment in which the system must operate changes. The people who use the system and the people who are served by the system also change over time.

G.5 REQUIREMENTS DEFINITION

G.5.1 PREREQUISITES TO SELECTING AN ARCHITECTURE

Before delving into the various architectures and tradeoffs, it is necessary to acknowledge that requirements definition is the primary key to a successful implementation of any system. Even before physical designs consider architectural tradeoffs, the requirements must be thoroughly defined, thought through, and clearly articulated. The graphical, client/server, complex-function systems in use today are often more useful but also far more difficult and expensive to build than the simpler systems of only a few years ago. A modern graphical system can be more flexible, responsive, and easier to use than the process-oriented, character-based systems that were, until recently, far more common. They can also provide significantly better services. However, such systems are more complex and more difficult to build than an “equivalent” character-based system. With reasonable application of certain principles, however, it is possible to avoid or minimize most of the pitfalls of implementing this type of application.

Contrary to some popular opinion, graphical systems are not inherently better for all purposes and needs than character-based systems. There are purposes for which graphical interfaces are clearly the better approach. There are purposes for which character-based systems are without question a better approach.

To illustrate with an example, consider these two activities:

1) Choosing an item from a pre-defined list

This is clearly best handled with a graphical approach wherein the short list of possible answers is presented and the user simply selects the correct response from the list. (Usually by “clicking” on it, though other means can be equally effective.) To require the user to key-enter the correct selection from a printed list is wasted effort with a high probability of error. This

would create a lack of uniformity of data that would make it difficult to run queries against the data and result in less accurate query results.

2) *Entering an address*

This, however, would be silly to present as a graphical interface. Picture a screen image of a typewriter keyboard. Use the mouse to move the cursor over the “3” key. Click on it. Now move the cursor to the “7”-key image, click on it. And so on until you gradually enter 3724 Elderberry Drive. In this case, a graphical interface would be counterproductive.

While extreme, these examples illustrate the issues that occur throughout a systems design effort. Attempting to begin the design effort, or worse yet, the development effort, before the requirements of the system are clearly understood and concisely documented is the quickest and most painful path to expensive failure.

It is important that the “what,” the “requirements,” of a system be clearly defined before any subsequent steps are begun. It may sometimes be argued that external constraints must be dealt with first. This argument is correct, but only to the extent of ensuring that the external constraints are adequately and completely defined as requirements, and not by attempting to start the design because “part of it is pre-determined.” For example, there could be a law or policy dictating that all communications networks must be established via a dial-up process using “off-hours” timing. This must be listed as a fundamental requirement of the system.

G.5.2 WHAT CONSTITUTES USEFUL “REQUIREMENTS?”

Note that merely listing a set of statements as “the requirements” is not adequate. The requirements must be analyzed to ensure that they are mutually consistent. At an absolute minimum, there must not be unresolved conflicts between requirements. For example, it is unacceptable to have the above “off-hours” communications requirement and also have a requirement that “all updates must be posted from remote sites to the central system within 4 working hours.” Since the first “requirement” dictated that there would be at least a 9-hour wait during business hours, that is in direct conflict with the second requirement for posting

data within 4 hours. If these types of conflicting requirements are allowed to exist, then only four outcomes are possible:

- 1) Project will fail because it is impossible to meet the requirements.
- 2) Potential vendors will “no bid” because they recognize the impossibility of completing the project.
- 3) Only “responsive” bidders will be vendors who were not competent to recognize the problem.
- 4) Time and money will be wasted correcting the problem.

The requirements should also be analyzed to ensure they are cohesive. They must all support the purpose of the system and waste no time or effort on things that are not part of the desired system. Failure to do this is less likely than outright conflict to cause project failure, though the likelihood of failure is increased by such an omission. However, if the requirements are not cohesive, then, at a minimum, it is a guarantee that time and cost will be wasted on items that are not essential to project success.

Finally, it is highly recommended that the requirements be organized into related modules. Failure in this area is not likely to cause project failure, but success in this area is a significant help in mutual understanding among all involved in the project and significantly increases the likelihood of a timely and cost-effective completion of the project. Only after work on requirements is complete can later stages such as design or looking at architectures be undertaken.

No large project today can get to the end of a job without automated tools; CASE is just one such tool. A CASE tool, which helps in requirements definition, data modeling, and other system development functions, is purchased with the expectation of achieving a single integrated repository for all project aspects. Even though there are up-front cost implications of doing this, the payoff comes in a shorter development time, better testing activities, and most importantly, reduced maintenance costs for the resulting system. Maintenance of the system not only deals with the fixing of problems but also making changes and adding functional enhancements. While there are problems and limitations with the various CASE tools, the simple fact is that CASE tools do work and their outputs

(documentation, database schema, data models, application models, etc) are invaluable aids to understanding the relationships of:

- Requirements to processes, data, and other objects
- Requirements to testing
- Entities
- Models

There are additional benefits derived from having an integrated and automated tool when it comes to testing. For example a requirements-to-test procedure cross-reference matrix is and invaluable when it becomes time to structure a testing program because it insures that all requirements are actually tested. The RTM included in Appendix A is an example of a tool that can be used to perform this mapping.

The following subparagraphs describe considerations and lessons learned about CASE tools.

G.5.3 WHY A CASE TOOL?

From a development standpoint, here are the key features and benefits that can be derived from using a CASE tool:

- The results of almost all of the analysis result and much of the design can be captured.
- The tool provides the developer with significant assistance in ensuring the integrity and consistency of the application.
- Unlike paper or other disparate documentation, it provides a common, current source of information to everyone working on the project.
- A critical, and often overlooked, function of any CASE tool is the mapping or modeling of the application structure. An application is not just a collection of low-level functions. To be effective, an application must be an organized assembly of related functions. This organization (or “Structure” or “Hierarchy”) must be defined and understood “on paper.” Thus, an Application Model is an abstract representation of the actual application. Absent an application model, there is no way to organize the actual application into a coherent and integrated whole.

CASE Tool Considerations

To be successful with CASE, States should know:

- What your business is – the type of software to be developed, the importance of databases, graphical user interfaces, and operating systems constraints.
- What your development and your target system is. Is there a constraint imposed for the use of CASE tools?
- Whether there is a path from one system to the other – will there be a need for a gateway between the development systems and the target/production system.
- What types of documentation is needed – must there be a way to produce standard list-type documents or graphical and textual documents.
- Is there an integrated CASE tool environment – are there upper and lower CASE tools with defined interfaces, a proofed way to generated documentation, LAN access for team members, etc.
- Where to get good education and coaching.
- Where good experiences were made using CASE.
- Which failures were made using CASE.
- Whether management will follow the required shift in paradigms.

Additionally, the benefits and cost increases are often regarded in short term revenues and not in calculating the long-term benefit results (less rework following test and less bug fixing during the maintenance phase). Thus, a convinced management is absolutely a pre-condition for using CASE.

The keys to a successful use of CASE tools are based on human facts:

- A willing and conscious team (customer and vendor). There needs to be a balance of people who are convinced of the “new” way to produce software. Each team member must be able to articulate his/her thoughts and take courage from the team.
- The team should be conscious that they are “driving a new vehicle,” that they will face unknown problems.
- A protecting and promoting management that defends attacks and is sure of the team’s success with CASE in their project.

- Skilled and coached team members are indispensable for success with a CASE project.

Obstacles for CASE projects emerge from the following:

- Lack of information or incomplete information about CASE methods and tools.
- Not enough examples of success to lower the barrier against new approaches.
- Unrealistic expectations from CASE usage.
- Missing education or advising.
- Usage of the tool without knowledge about supported methods and techniques.
- Provincial or traditionalist mindset as opposed to a “pioneer” mindset.
- Wrong understanding at the management level.

Evaluating CASE Tools

Regarding the experiences and consideration discussed above, certain evaluation criteria must be established. The following points or a subset thereof, will be important:

- How can a potential CASE user become familiar with a chosen methodology? What methods are included?
- Is there support available and can the tool be easily modified?
- Does the tool fit the hardware and operating system requirements?
- Does the tool support LAN and multi-user environments?
- Does the tool provide a documentation generator?
- Are there references by customers and the methodology experts?
- Is there an open architecture for customizing purposes? Can you implement your own method or to add/change some rules of techniques to the existing set, or to take into account certain company guidelines, or the integration of accompanying tools (configuration management tools, etc.) that are not within the CASE tool?

- Is there a gateway to other CASE tools, e.g. to lower CASE tools such as code generators, if it is not included, of if you want to generate another language or in another environment.
- Are there animation or simulation facilities available?

CASE Tool Is NOT Part of the Architecture

It is important to note that the selection of a CASE tool is entirely independent of existing or planned architectures. The CASE tool should be used to develop the requirements for the planned system, which must be done before the architecture is selected. The selection of an architecture may drive the selection of the supporting hardware infrastructure. That infrastructure may well be entirely different from whatever infrastructure is used to support the CASE tool.

G.6 SYSTEM ARCHITECTURAL CONSIDERATIONS

Any candidate architecture must satisfy the requirements and boundary conditions of the system. Those requirements and conditions are detailed in the Logical Design Document (LDD). Once the logical design is completed, an application architecture approach is considered.

After the application architecture is selected, the physical alternatives - the software tools, operating systems, and hardware - necessary to develop and implement the system are examined. The analysis follows parallel paths because much of the analysis of the software implementing components (Relational Database Management System-RDBMS- and operating systems) influences the selection of the development tools and vice versa. The RDBMS alternatives need to be examined first because the RDBMS is the key performance component in the application. Once the RDBMS is selected, it then drives (or at least limits) the selection of the server operating system, which in turn drives the selection of the client operating system. Finally, the hardware alternatives are examined.

There are two basic types of system architectures: centralized and distributed (de-centralized). Table I lists the advantages and disadvantages of each. Either the data, the system processes, or both may be distributed or centralized. These issues must be considered separately for data and processes. In some

circumstances it may be valid for one to be distributed while the other is centralized.

There are two basic types of system architectures: centralized and distributed (de-centralized). Table I lists the advantages and disadvantages of each. Either the data, the system processes, or both may be distributed or centralized. These issues must be considered separately for data and processes. In some circumstances it may be valid for one to be distributed while the other is centralized.

Table I. Centralized vs. Distributed Architectures

Architecture	Advantages	Disadvantages
Centralized	Centralized maintenance of database and application Less complex Single instance of data elements, no collisions Lower initial costs to design, develop, and implement Growth facilitation – can simply add “more power” up to the maximum capacity limits	There are multiple “Single points of failure” (several things, any one of which can disable the entire system) Contention and other performance issues Requires either an on-line, real-time network capability (very expensive) or acceptance of significant periods of unusable status Much higher on-going operational costs
Distributed	Data and/or processes located near the “greatest demand” Faster data access (potentially) Faster data processing Growth facilitation – less likely to encounter maximum capacity limits No single point of failure, greater system reliability Can be implemented in areas where reliable communications networks are either unavailable or prohibitively expensive Lower operational costs	Complex management and control of distributed data/process Security is less sophisticated and more difficult to implement Data may be valid, but not timely Multiple instances of data elements, collisions must be handled Higher initial costs to design, develop, and implement

New WIC systems can take advantage of a distributed architecture mainly because that is where the greatest advantages in technology have occurred. In

some cases, it is the only feasible means of implementation that does not require prohibitive investment.

Where highly reliable, on-line, real-time communications networks of adequate bandwidth are available at reasonable costs, either centralized or distributed architectures may be used. If any of those conditions cannot be met, then either a distributed architecture must be used, or frequent and extended periods of partial system unavailability must be acceptable.

Note that an effectively designed distributed system can be effectively used in a centralized network architecture, if desired. The reverse is NOT true (a centralized system cannot be used in a distributed network architecture).

It is worth noting the two variations of a distributed architecture:

- 1) Distributed processing system – processing of data is shared between two or more locations. Data is stored in a single, central site. Also known as Multiple-Site Processing, Single-Site Data. A typical architecture requires the use of a network file server on which conventional applications are accessed through a LAN. There is no relationship between whether a system is client/server and whether it is distributed or centralized.
- 2) Distributed database system – data is stored over two or more physically independent sites. Database can be one or many logical databases. Processing may or may not be distributed (but as a practical matter, it almost always is at least partially distributed.) Architecture contains multiple data processors and usually multiple transaction processors at multiple sites. This approach is sometimes referred to as Multiple-Site Processing, Multiple-Site Data. Two variations of the architecture depend upon whether the database system is homogenous (integration of a single type of database) or heterogeneous (integration of different types of databases).

G.6.1 REQUIREMENTS AFFECTING ARCHITECTURE SELECTION

Table II lists typical requirements of a system that affect the choice of an architecture and its physical components (user interface development tool,

database engine, network operating system, client operating system, communications, and hardware).

G.7 SUMMARY

It is important to re-emphasize that prior to selecting an architecture, requirements must be clearly and concisely defined, with no unresolved conflicts. The project team (customer and vendor) for a complex system, such as required by today's WIC requirements, should give serious consideration to using a CASE tool in at least documenting the requirements and the design. Once the requirements have been clearly documented, then and only then can architectural tradeoffs and selection begin. The operational environment heavily influences the selection of an architecture; hence, there is no "best" architectural selection for a WIC system. In some cases, it is possible to combine architectures, building an overall system that uses a "best of breed" approach to solve multiple, concurrent problems.

Table II. Requirements and Their Impacts Upon Architecture

Requirement	Impact
The application must be capable of independent, stand-alone operation in any clinic location for up to 24 hours without communication to any other site or the state office.	Data replication and coordination after the resumption of communications must be fully automatic. (Absolutely requires asynchronous, distributed architecture. Not only will centralized architecture not work, but any architecture requiring coordinated updates of data will also not work. Note that a system implemented in this fashion is capable of operating in ANY architecture. This is the most flexible of all designs, and correspondingly the most expensive to design and develop. Its expense in on-going operations will depend on the environment in which it is actually implemented.)
The system architecture must be focused on simplicity and ease of support/maintenance.	This will push very strongly for a totally centralized system. It may totally rule out other approaches if reliable and inexpensive real-time communications of adequate bandwidth is available.
Note that the two previous "requirements" are essentially mutually exclusive.	They both sound like good ideas, maybe even on a par with "Mom and Apple Pie." However, they actually force the system into two completely different directions. While it is possible to implement a system with an architecture that pretty much supports both requirements, the costs of doing so are exponentially higher than the costs for doing either one, alone.

Requirement	Impact
Each element of data is to have one and only one “owner” location.	That owner is responsible for any maintenance of the data element, as well as publishing changes in that element to other systems as appropriate. This has absolutely no impact on central vs. distributed unless there is a corresponding requirement that data be maintained separately at the owner location and published elsewhere only as needed. That would tend to, but not require, a distributed approach.
An individual’s security and access authorizations are to follow that person wherever s/he goes among the application site locations.	Can be implemented either way.
The system is to be built around a Graphical User Interface (GUI) architecture.	No impact.
Data is to be public between locations; publication is to be an automated function.	No impact. Note that the use of terms like “Publication” which is a distributed database term, may imply an architecture direction that is not actually intended in the requirement. To correct this, change the word “Publication” to the words “Reporting” or “Data Availability,” and any preference to central or distributed architecture also disappears. Be careful not to use “code words” that could unnecessarily influence your decision process.
Single application design and single database design.	No impact.
Generation of reports at all levels.	No impact.
Adherence to standards.	No impact.
Fit within the current physical locations and communications infrastructure.	Could drive selection in either direction depending on current infrastructure.
Centralized or de-centralized support of the system components.	No impact. But likely part of an organization that already tends to distributed systems and will tend to continue that approach.
Scalability of the database.	No impact.
Simple, fast, and efficient distribution of application and database changes and upgrades.	Strong driver towards centralized architectures.